5流量控制和连接管理

2019年6月18日 20:39

-
- TCP的流量控制

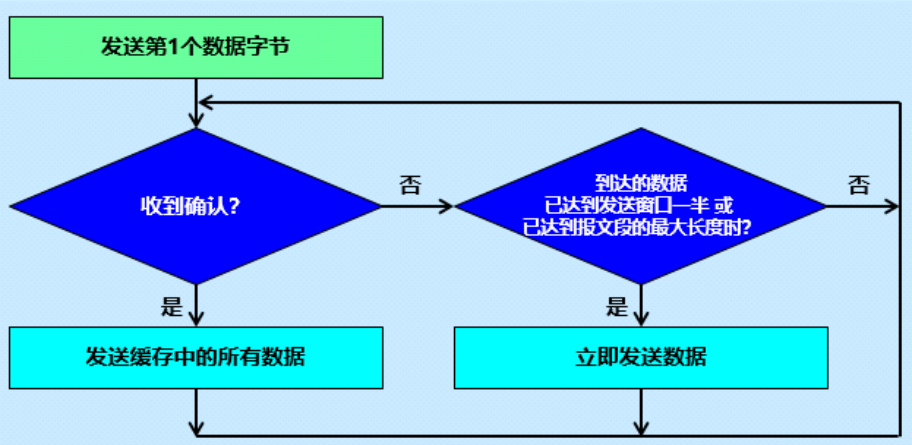
1- 滑动窗口实现流量控制

1. flow control流量控制：让发送速率不要太快，让接收方来得及接收
   (1) 注意窗口单位是字节，不是报文段序号
   (2) ACK表示确认位，ack表示确认字段值，rwnd表示接收窗口大小
2. 接收方B接收缓存不够了以后，即可向发送方A更新rwnd=0的报文
   (1) 这个报文称为零窗口通知
   (2) A收到后也会立刻停止继续发送
   (3) 不过发送零窗口通知后也有需要接收的临时报文：零窗口探测报文，确认报文，紧急数据报文段
3. B上交了一定量的数据后，即可向A更新rwnd为新的较大值的报文
   (1) 若之前给了A零窗口通知，现在这个更新rwnd的报文却丢失了，则会进入死锁局面，A不敢发，B收不到
   (2) 为解决死锁的可能，TCP为每个连接设置一个persistence timer持续计时器，定时“探测”：发送一个仅携带1字节数据的零窗口探测报文段，要求对方更新rwnd值

2- TCP的传输效率

1. 发送报文的时机
   (1) 当缓存内字节数达到MSS时就组装成一个报文段并发送
   (2) 当对面发来PSH请求时，直接发送
   (3) 发送计时器的期限到了，将缓存内数据装入报文段发送
2. Nagle算法
   (1) silly window syndrome糊涂窗口综合症：RFC813：接收窗口字节数很少时，信道利用率和有效数据传输效率变得极低
   (2) Nagle算法：发送方第一次只送一个字节，缓存后序的所有数据，当收到确认报文后再将缓存全发送出去；之后也是，收到前一个报文的确认后才发下一个报文段；接收方的接收缓存到达swnd的一半或已到达MSS时也可以直接发确认报文，通知对方可以发送下一个报文段

(3)

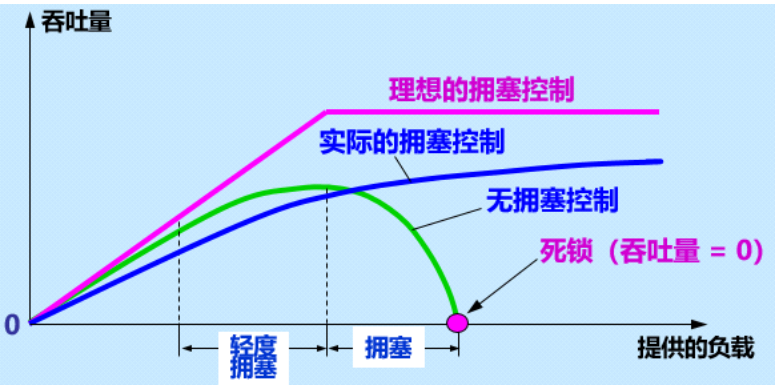


-
- TCP的拥塞控制

1- 拥塞控制的一般原理

1. congestion拥塞：总要求资源>可用资源
2. 产生原因：缓存容量小；链路容量小；处理机处理速率小；拥塞本身加剧拥塞
3. 拥塞控制：防止过多数据注入网络，使链路和路由器不过载；是全局管理
   (1) 流量控制：防止接收端来不及接收；是端到端的通信量控制问题

(2)



   (3) 上图横轴offered load又称输入负载或网络负载，表示单位时间内输入进网络的分组数目
   (4) 纵轴是throughput吞吐量，代表单位时间内从网络输出的分组数目
   (5) 拥塞是动态的问题，拥塞控制本身甚至能引起网络性能恶化甚至死锁
4. 解决方法
   (1) Q：增加资源能解决吗
   (2) A：不能。盲目增大缓存而不增处理速度可能使排队等待时间增加，引起大量超时重传；只提高处理机速率会将瓶颈转移去他处
   (3) **开环控制**：设计网络时力争考虑周全，不发生拥塞
   (4) 闭环控制：基于**反馈环路**，按网络运行状态采取相应控制措施
5. 闭环控制措施
   (1) 监测网络系统，检测拥塞在何时何处发生
      1) 主要监测：缺缓存而丢弃分组的比例；平均队列长；超时重传分组数；平均分组时延；分组时延的标准差
   (2) 将拥塞发生的信息传送到可采取行动的地方
      1) 传递拥塞通知：通知拥塞发生的分组；在分组中保留表示拥塞状态的字段；周期性地发出探测分组

(3) 调整网络系统的运行以解决出现的问题

1) 调整过于频繁会使系统产生不稳定的振荡；迟缓采取行动又无价值

2) 调整思路：增加网络可用资源；减少用户对资源的需求

2- TCP的拥塞控制方法RFC5681

1. congestion window拥塞窗口cwnd

(1) 慢开始和拥塞避免是两个基于cwnd的拥塞控制

(2) 发送方让自己的**发送窗口=拥塞窗口**

1) 严格地说，发送窗口上限=min{对方的接收窗口，自己的拥塞窗口}

2) 根据网络拥塞程度，动态改变拥塞窗口大小

3) 按现在通信线路的质量，因传输差错而丢弃分组的概率<1%

4) **判断拥塞的依据是出现超时**，超时指一直收不到某序号的确认

5) 判断出拥塞后，一般立即设置拥塞窗口**cwnd=SMSS**

2. slow-start慢开始：cwnd值从小逐渐增大，试探网络负载能力

(1) cwnd每次增量=min（N，SMSS）其中N是原先未被确认的、但现在被刚收到的确认报文段所确认的字节数（应该是与首部ack有关）；SMSS是Sender Maximum Segment Size发送方最大报文段

(2) cwnd初值不推荐超过2~4个SMSS或超过6570字节

(3) cwnd/SMSS=每一transmission round传输轮次可发送的报文数

(4) 实际应用上，每次收到了B的确认报文，A就可多发1个报文，即让cwnd增加1SMSS

(5) 一般来说，每经过一传输轮次，cwnd都会翻倍

(6) 所以cwnd增速极快，“慢”开始并不是指增速慢，其实是指刚开始发送的报文很小，不会突然给网络增加很多负担

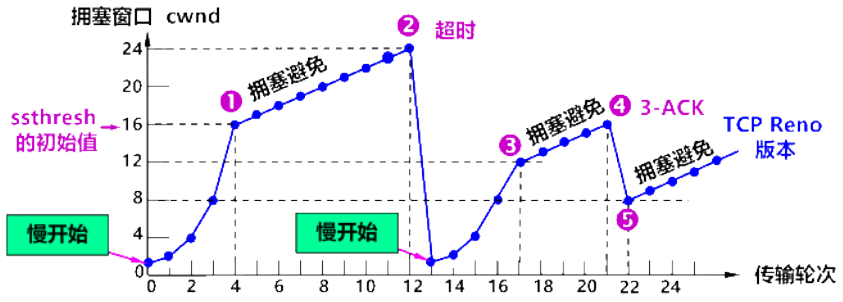
3. congestion avoidance拥塞避免：cwnd值线性缓慢增长

(1) 每次收到确认报文，就让cwnd增加1SMSS*SMSS/cwnd，即多发SMSS/cwnd个报文，则一个传输轮次下来，cwnd恰增加了1SMSS

(2) additive increase**加法增大**AI，拥塞避免依旧会增加拥塞的概率，此处“避免”只是指长得慢，不易出现拥塞

(3) 下图纵轴单位为SMSS，ssthresh初值为16个SMSS，Reno是指新版本，区分Tahao版本

(4)



当 TCP 连接进行初始化时，将拥塞窗口置为 1。图中的窗口单位不使用字节而使用报文段。

4. ssthresh慢开始门限：确认是否用慢开始算法更改cwnd的门限

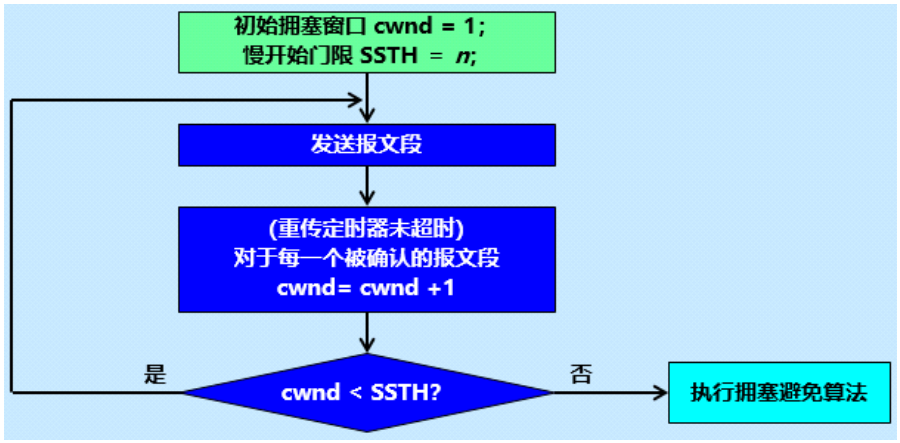
(1) cwnd<ssthresh时用慢开始算法，快速增长cwnd

(2) cwnd>ssthresh时用拥塞避免算法，缓慢增长cwnd

(3) cwnd=ssthresh时两个都能用

(4) 发现超时未确认时，令**ssthresh=max（cwnd/2，2）**

(5)



5. fast retransmit快重传：B发现个别报文段丢失后，立即向A发送确认报文

(1) A接到该确认后，会立即进行重传，实现快重传

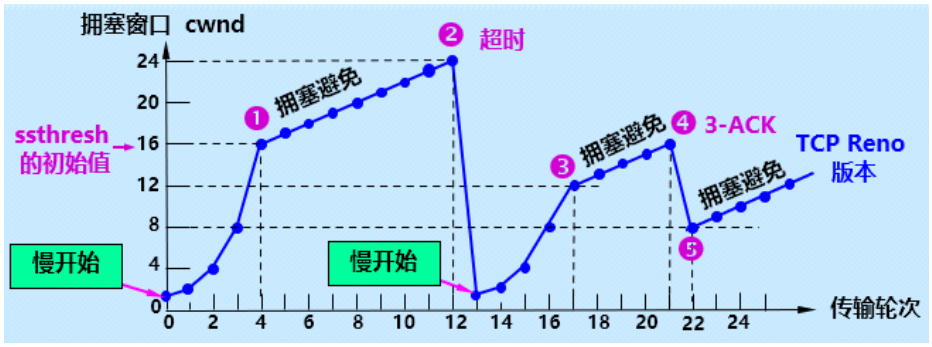
(2) A连续收到3个对同一序号的ACK，说明该序号后面的一个序号丢失或迟到的情况下，更后面的连续三个序号都到达了，因此需要快速重传

6. fast recovery快恢复：令**cwnd=ssthresh=cwnd/2**

(1) 用于检测到重复三连ACK时，说明需要快重传，但不一定是出现了拥塞，因此降低一半cwnd，再用加法增大试探

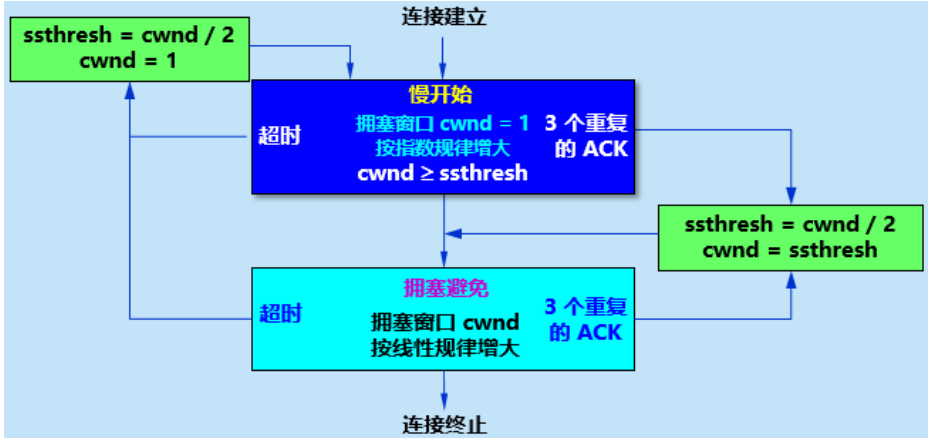
(2) 这个除以2被称作multiplicative decrease**乘法减小**MD

(3)



(4) AI和MD统称为AIMD算法

(5)



3- Active Queue Management主动队列管理AQM

1. 传统的TCP报文丢弃策略：先进先出FIFO，tail-drop policy尾部丢弃

(1) 丢弃路由器队列尾部往往会导致一连串分组的丢失，迫使发送方超时重

传，再慢开始

(2) global syncronization全局同步：许多TCP连接同时进入慢开始状态，使全网通信量急降，网络恢复正常后再突增

(3) 98年提出来主动队列管理，当队列长度到达某一值得警惕的数值时，主动丢弃分组

2. Random Early Detection随机早期检测RED（D还可以是Drop或Discard）

(1) 当网络中平均队列长度<最小门限，则直接排队；若>最大门限，则直接丢弃；若处于之间，则按概率p丢弃

(2) 2015年的RFC7567已不推荐RED，但AQM还是需要的

-
- TCP的运输连接管理

1. TCP连接三个阶段：连接建立、数据传送、连接释放

1- TCP的连接建立

1. 建立连接的三个问题

(1) 双方互相确认对方存在

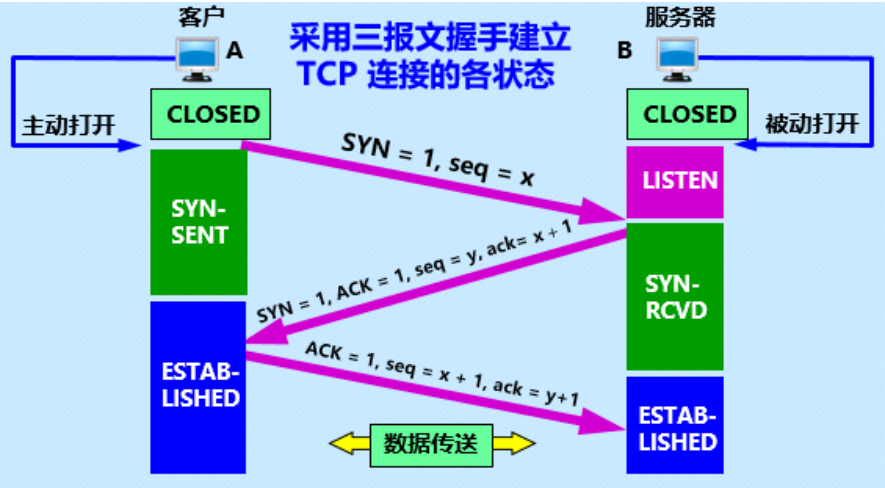
(2) 协商参数（窗口最大值，是否使用窗口扩大、时间戳选项、服务质量）

(3) 对运输实体资源进行分配（缓存大小，连接表中的项目等）

2. 连接建立：三报文握手

(1) TCP连接的建立采用客户服务器方式，主动发起连接的A视作client客户，被动等待连接建立的B视作server服务器

(2) 三报文主要是为了**防止迟到的连接请求引起不必要的连接建立**

(3) 

(4) 客户A先主动发出请求（SYN=1），并进入SYN-SENT状态

(5) LISTEN状态的B的服务器进程收到该报文后，创建一个Transmission Control Block传输控制块TCB，再发送同样SYN=1的第二条报文，进入SYN-RCVD状态

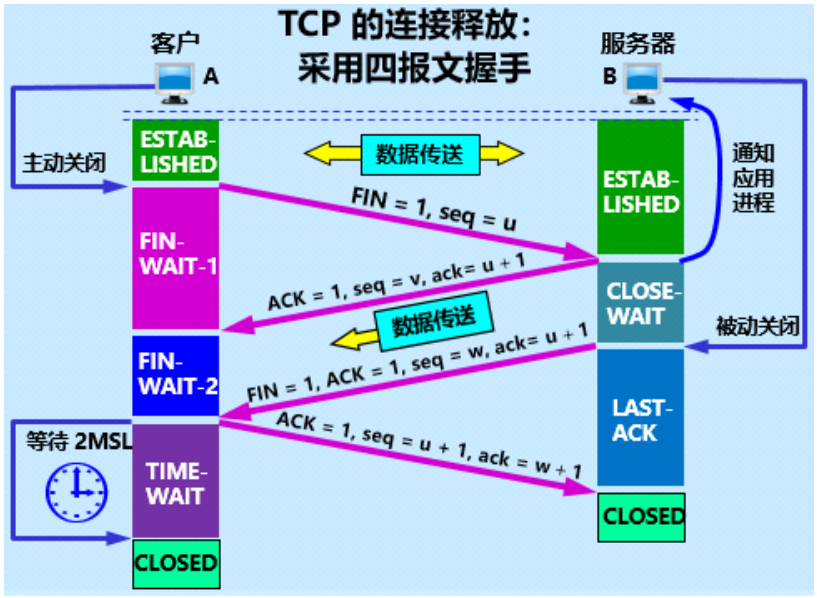
(6) A在收到这第二个报文后进入ESTABLISHED，并立刻发送第三条报文（ACK=1）；B通过第三条报文确认第二条报文被A收到后，也正式ESTABLISHED

(7) 第二条报文同时负责了确认收到第一条报文（ACK=1，ack=x+1）和请求对方确认无误（SYN=1，seq=y）两条功能，可以拆成两次发，成为四报文握手

2- TCP的连接释放

1. TCP规定释放连接请求报文段的FIN=1，且该报文必须消耗一个序号

(1)



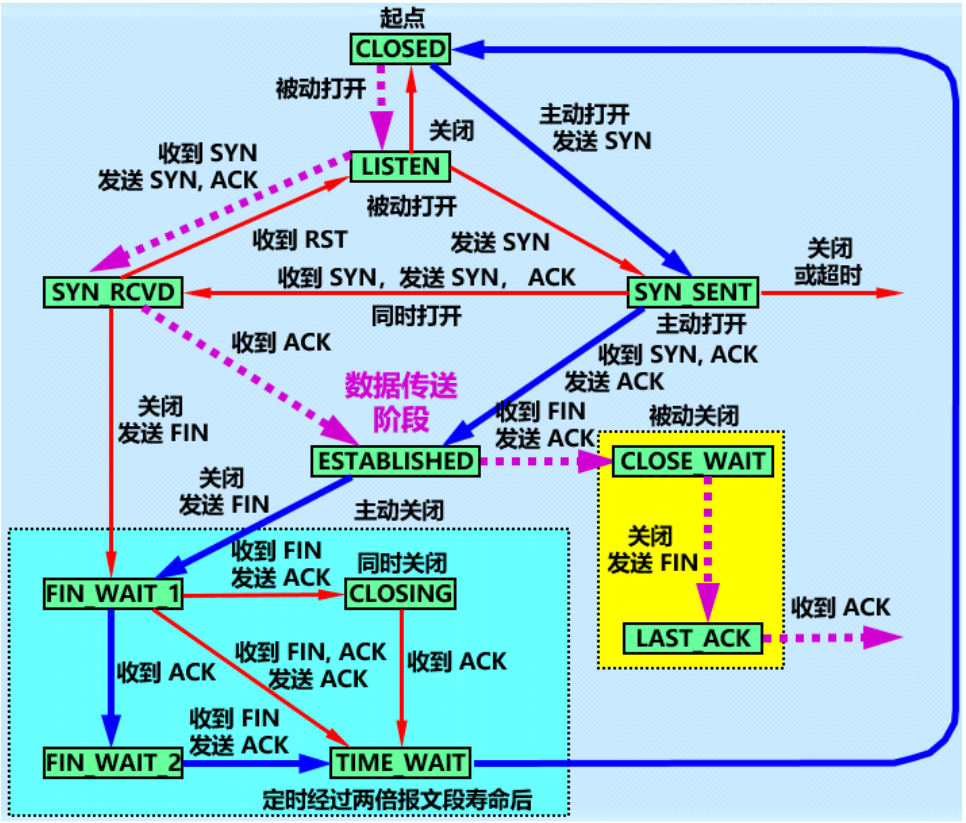
2. 释放连接的四报文
   (1) A先发送释放请求（FIN=1，seq=u，u-1为A之前发的最后一个数据字节的序号）并进入FIN-WAIT-1状态，等待B的确认
   (2) B收到第一条报文后，先给A发送确认（ACK=1，seq=v，ack=u+1，v-1为B之前发的最后一个数据字节的序号）并进入CLOSE-WAIT状态，并通知上级应用进程
   (3) A收到B发来的请求后进入FIN-WAIT-2状态，等待B也打算关闭连接
   (4) B的进程确认没有想发给A的数据时，发送释放请求（FIN=1，ACK=1，seq=w，ack=u+1，w-1是B之前发的最后一个数据字节的序号，u是A的释放请求的序号）并进入LAST-ACK状态，等待A确认关闭
   (5) A收到后发出确认（ACK=1，seq=u+1，ack=w+1）进入TIME-WAIT状态，等待2Maximum Segment Lifetime时间后正式CLOSED，并撤销相应的TCB
   (6) B也在收到该请求后撤销TCB，释放连接
3. 等待2倍MSL时间的原因
   (1) **保证第四条报文段不丢失。**这是通过B迟迟收不到确认时会发重传第三条报文来实现的，A收到这个重传时，重新发第四条报文并重启计时
   (2) 确保网络中没有迟到的报文，如失效的连接请求报文，为防止它影响下次连接，多等一会
4. keepalive timer保活计时器
   (1) 客户端主机突然故障，没必要继续保持连接，但又发不出释放连接请求
   (2) 需要服务器自动判断有没有必要主动断开连接
   (3) B每收到A一次报文，就重启一次计时器，若计时器达到0，就每隔75秒发送一探测报文段，连续发送10个后若仍无响应，则主动断开连接

3- TCP的有限状态机

   (1) 粗实线表示客户进程正常变迁
   (2) 粗虚线表示服务进程正常变迁

(3) 细线表示异常变迁

1.



(1) 细线异常可能和分组丢失有关，分组丢失除了在拥塞状况下会发生外，还在这些情况下发生：

1) IP层某分片迟到，终点来不及组装，全部丢失
2) 到达终点，终点缓存不够了
3) 路过某局域网的网桥，网桥空间不够了
4)
5)
6)
7) ----------我是底线----------